com.geps.util.ldap

Class BaseDirectoryAdapter

```
java.lang.Object
  |
  +--com.geps.util.ldap.BaseDirectoryAdapter
```

---

public abstract class **BaseDirectoryAdapter**
extends java.lang.Object
implements IBaseObjectClass

Class Description:

Base class for all Directory Adapters. This class is abstract and cannot be instantiated.

Field Summary

protected DirectoryEntry	**m_dirEntry**
protected java.util.ArrayList	**m_modifications**

Constructor Summary

BaseDirectoryAdapter()

Method Summary

DirectoryEntry	**getDirEntry**() Desc: Use to get the DirectoryEntry from the adapter.
java.util.ArrayList	**getModifications**() Desc: Use to get the list of ModificationItem(s) applied to the adapter.
protected void	**initialize**(DirectoryEntry de) Desc: Used to initialize the adapter.

Methods inherited from class java.lang.Object
, clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Field Detail

m_dirEntry

```
protected DirectoryEntry m_dirEntry
```

---

m_modifications

```
protected java.util.ArrayList m_modifications
```

Constructor Detail

BaseDirectoryAdapter

```
public BaseDirectoryAdapter()
```

Method Detail

initialize

```
protected void initialize(DirectoryEntry de)
                  throws javax.naming.NamingException
```

Desc: Used to initialize the adapter. This is used by com.geps.util.ldap.DirectoryManager when its getAdapterInstance() method is called.

Parameters:

de - DirectoryEntry to initialize adapter with.

---

getDirEntry

```
public DirectoryEntry getDirEntry()
```

Desc: Use to get the DirectoryEntry from the adapter.

Specified by:

getDirEntry in interface IBaseObjectClass

Returns:

Returns the DirectoryEntry associated with the adapter.

---

getModifications

```
public java.util.ArrayList getModifications()
```

Desc: Use to get the list of ModificationItem(s) applied to the adapter. This method should not be used by clients. It is used by com.geps.util.ldap.DirectoryManager.

Specified by:

getModifications in interface IBaseObjectClass

Returns:

Returns the list of ModificationItem(s).

---

com.geps.util.ldap

Class DirectoryEntry

```
java.lang.Object
  |
  +--com.geps.util.ldap.DirectoryEntry
```

---

public class **DirectoryEntry**
extends java.lang.Object

Class Description:

Simple wrapper which represents a DirContext.

Field Summary

private java.lang.String	m_dn
private javax.naming.directory.DirContext	m_entry

Constructor Summary

DirectoryEntry(javax.naming.directory.DirContext entry) Desc: Constructor.

Method Summary

javax.naming.directory.DirContext	getDirCtx() Desc: Use to retrieve the entry.
java.lang.String	toString() Desc: Override toString().

Methods inherited from class java.lang.Object

, clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, wait, wait, wait

Field Detail

m_dn

```
private java.lang.String m_dn
```

---

m_entry

```
private javax.naming.directory.DirContext m_entry
```

Constructor Detail

DirectoryEntry

```
public DirectoryEntry(javax.naming.directory.DirContext entry)
```

Desc: Constructor.

Method Detail

getDirCtx

```
public javax.naming.directory.DirContext getDirCtx()
```

Desc: Use to retrieve the entry.
Returns:
Returns the entry.

---

toString

```
public java.lang.String toString()
```

Desc: Override toString().
Overrides:
toString in class java.lang.Object

---

com.geps.util.ldap

Class DirectoryManager

```
java.lang.Object
  |
  +--com.geps.util.ldap.DirectoryManager
```

---

public abstract class **DirectoryManager**
extends java.lang.Object

Class Description:

This class represents the Directory Framework in which clients will interface with to request DirectoryEntry and object class adapters and to write entry modifications back to LDAP.

Field Summary

private static java.lang.String	**s_adapterPkg**
private static javax.naming.directory.InitialDirContext	**s_ctx**

Constructor Summary

DirectoryManager()

Method Summary

(package private) static void	()
private static java.lang.String	**extractLdapObjClassName**(java.lang.String name) Desc: Helper which extracts the object class name from the specified 'name'.
static BaseDirectoryAdapter	**getAdapterInstance**(DirectoryEntry entry, java.lang.String adapterName) Desc: Use to obtain the specified 'adapterName' adapter from the specified 'entry'.
static java.util.ArrayList	**getAdapters**(DirectoryEntry entry) Desc: Use to obtain a list of all adapters that the specified 'entry' is composed of.
static DirectoryEntry	**getEntry**(IBaseObjectClass adapter) Desc: Use to get DirectoryEntry from the specified 'adapter'.
static DirectoryEntry	**lookup**(java.lang.String dn) Desc: Retrieves the DirectoryEntry whos key matches the the specified 'dn'.
static java.util.ArrayList	**search**(java.lang.String ctxToSearch, java.lang.String filter) Desc: Use to execute a query against the Directory.

static void	write(IBaseObjectClass adapter) Desc: Use to write out the contents specified by 'adapter' to LDAP.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Field Detail

s_ctx

```
private static javax.naming.directory.InitialDirContext s_ctx
```

---

s_adapterPkg

```
private static java.lang.String s_adapterPkg
```

Constructor Detail

DirectoryManager

```
public DirectoryManager()
```

Method Detail

lookup

```
public static DirectoryEntry lookup(java.lang.String dn)
                             throws javax.naming.NamingException
```

Desc: Retrieves the DirectoryEntry whos key matches the the specified 'dn'.

Returns:

DirectoryEntry associated with the specified 'dn'.

Throws:

javax.naming.NamingException - If a naming exception occurs or if lookup did not return object of type DirContext.

---

getAdapterInstance

```
public static BaseDirectoryAdapter getAdapterInstance(DirectoryEntry entry,
                                                      java.lang.String adapterName)
                                               throws javax.naming.NamingException,
                                                      java.lang.ClassNotFoundException,
                                                      java.lang.InstantiationException,
                                                      java.lang.IllegalAccessException
```

Desc: Use to obtain the specified 'adapterName' adapter from the specified 'entry'. If the requested 'adapterName' is not an object class of 'entry', a null will be returned.

Parameters:

entry - DirectoryEntry in which to search

Returns:
The adapter representing the object class specified by 'adapterName' from the DirectoryEntry 'entry'. null is returned if the specified 'adapterName' is not an object class of the specified 'entry'.
Throws:
javax.naming.NamingException - If a naming exception occurs.

---

getEntry

```
public static DirectoryEntry getEntry(IBaseObjectClass adapter)
```

Desc: Use to get DirectoryEntry from the specified 'adapter'.
Parameters:
`adapter` - The adapter to get the DirectoryEntry from.
Returns:
The DirectoryEntry for the specified 'adapter'.

---

write

```
public static void write(IBaseObjectClass adapter)
                  throws javax.naming.NamingException
```

Desc: Use to write out the contents specified by 'adapter' to LDAP.
Parameters:
`adapter` - Object Class to write out.

---

getAdapters

```
public static java.util.ArrayList getAdapters(DirectoryEntry entry)
                                        throws javax.naming.NamingException
```

Desc: Use to obtain a list of all adapters that the specified 'entry' is composed of. Each adapter name returned can be passed into DirectoryManager.getAdapterInstance(DirectoryEntry, String) as the 2nd parameter to obtain an adapter instance.
Parameters:
`entry` - DirectoryEntry in which to discover all adapters for.
Returns:
Non-null ArrayList of String adapters names.

---

search

```
public static java.util.ArrayList search(java.lang.String ctxToSearch,
                                         java.lang.String filter)
                                  throws javax.naming.NamingException
```

Desc: Use to execute a query against the Directory.
Parameters:
`ctxToSearch` - Context to search. "" for current context.
`filter` - LDAP filter.
Returns:
List of DirectoryEntries resulting from the query. Only DirContext objects are supported so if the query returns objects other than DirContext, they will not be in the List.

---

extractLdapObjClassName

```
private static java.lang.String extractLdapObjClassName(java.lang.String name)
```

Desc: Helper which extracts the object class name from the specified 'name'. 'name' looks like "com.geps.ldap.PocuserAdapter". This method removes all package names the "Adapter" suffix is stripped and the remaining string returned.

Parameters:

name - Adapter names which are defined constants in DirectoryConstants or the object class name itself.

Returns:

Returns the LDAP object class name.

```
static void ()
```

---

com.geps.util.ldap

Class Generator

```
java.lang.Object
  |
  +--com.geps.util.ldap.Generator
```

---

public abstract class **Generator**
extends java.lang.Object

Class Description:

This class is used to generate java interfaces and adapters which represents LDAP object classes. For each LDAP object class there is one java interface and one java adapter. This class also generates the file DirectoryConstants which provides defined constants used to identify adapters. These classes \ are used in the java LDAP Directory framework. This class is abstract so it cannot be instantiated.

Inner Class Summary

private static class	Generator.NBP Desc: Helper class, Name Boolean Pair.

Field Summary

private static java.lang.String	s_copyRightYear
private static java.lang.String	s_dateGenerated
private static java.lang.String	s_DirConstName
private static java.lang.String	s_genSrcPath
private static javax.naming.directory.InitialDirContext	s_initDirCtx
private static java.lang.String	s_multiSuffix
private static javax.naming.directory.DirContext	s_schemaRoot
private static java.lang.String	s_srcPathRoot
private static java.lang.String	s_srcPkg
private static java.lang.String	s_ts

Constructor Summary

`Generator()`

Method Summary

(package private) static void	`()`
private static void	`addAttrNameToList(java.util.TreeSet store, javax.naming.NamingEnumeration vals)` Desc: Add attribute names to a TreeSet.
private static void	`emitAdapter(java.lang.String oc, java.util.TreeSet[] attrNames, java.io.PrintStream out)` Desc: Generates the adapter class for the specified object class.
private static void	`emitAdapterClassBody(java.lang.String oc, java.util.TreeSet[] attrNames, java.io.PrintStream out)` Desc: Writes the body of the adapter.
private static void	`emitAdapterGetters(java.util.TreeSet attrSet, java.io.PrintStream out)` Desc: For the specified 'attrSet' will generate getters for all attributes contained within for the adapter.
private static void	`emitAdapterImports(java.io.PrintStream out)` Desc: Writes the import statements for the Adapter.
private static void	`emitAdapterName(java.lang.String oc, java.io.PrintStream out)` Desc: Writes the adapter name and opening curley.
private static void	`emitAdapterSetters(java.util.TreeSet attrSet, java.io.PrintStream out)` Desc: Generate adapter setters for attributes.
private static void	`emitAdapterToString(java.util.TreeSet[] attrNames, java.io.PrintStream out)` Desc: Writes out the adapters toString() method.
private static void	`emitAdapterToStringHelper(java.io.PrintStream out)` Desc: Writes out the adapters toString() helper method.
private static void	`emitAllAdapterGetters(java.util.TreeSet[] attrNames, java.io.PrintStream out)` Desc: Writes out all adapter getters for both required and optional attributes.
private static void	`emitAllAdapterSetters(java.util.TreeSet[] attrNames, java.io.PrintStream out)` Desc: Writes out all adapter setters for both required and optional attributes.
private static void	`emitAllInterfaceGetters(java.util.TreeSet[] attrNames, java.io.PrintStream out)` Desc: Writes out all interface getters for both required and optional attributes.
private static void	`emitAllInterfaceSetters(java.util.TreeSet[] attrNames, java.io.PrintStream out)` Desc: Writes out all interface setters for both required and optional attributes.
private static void	`emitClosingClassBracket(java.io.PrintStream out)` Desc: Writes the class closing curley.
private static void	`emitCommentHeader(java.io.PrintStream out)` Desc: Writes the comment header for the file.
private static void	`emitDirConst(java.lang.String className, java.io.PrintStream out)` Desc: Write out adapter constant for the specified 'className'.
private static void	`emitDirConstName(java.io.PrintStream out)` Desc: Writes the interface name and opening curley.

private static void	**emitGetFromModifiedCache**(java.io.PrintStream out) Desc: Writes out getFromModifiedCache method.
private static void	**emitInterface**(java.lang.String oc, java.util.TreeSet[] attrNames, java.io.PrintStream out) Desc: Generates the interface class for the specified object class.
private static void	**emitInterfaceClassBody**(java.util.TreeSet[] attrNames, java.io.PrintStream out) Desc: Writes the body of the interface.
private static void	**emitInterfaceGetters**(ja. .util.TreeSet attrSet, java.io.PrintStream out) Desc: For the specified 'attrSet' will generate getters for all attributes contained within for the interface.
private static void	**emitInterfaceImp rts**(java.io.PrintStream out) Desc: Writes th import statements for the Interface.
private static void	**emitInterfaceName**(java.lang.String oc, java.io.PrintStream out) Desc: Writes the interface name and opening curley.
private static void	**emitInterfaceSetters**(java.util.TreeSet attrSet, java.io.PrintStream out) Desc: For the specified 'attrSet' will generate setters for all attributes contained within for the interface.
private static void	**emitPackage**(java.io.PrintStream out) Desc: Writes the package statement.
private static void	**generate**(java.lang.String[] objClasses) Desc: Directs the generation of interface and adapter classes.
private static void	**getAttributes**(java.lang.String oc, java.util.TreeSet mandatory, java.util.TreeSet optional) Recursively extracts all attributes for the specified 'oc' and all attributes of 'oc' superclasses.
private static java.util.TreeSet[]	**getAttrList**(java.lang.String oc) Desc: Create a list of required and optional attribute names for the specified 'oc' object class and all attributes of 'oc' super object classes and so on by looking up these values in the LDAP Schema.
private static java.lang.String[]	**getObjClasses**() Desc: This method will query LDAP to get all object class names and returns those names in an array of Strings.
private static void	**initialize**() Desc: Gets required system properties, figures out where to create the generated files.
static void	**main**(java.lang.String[] args) Desc: Program entry point.
private static void	**shutDown**() Desc: Release any remaing resources.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Field Detail

s_multiSuffix

```
private static final java.lang.String s_multiSuffix
```

s_ts

```
private static java.lang.String s_ts
```

---

s_DirConstName

```
private static final java.lang.String s_DirConstName
```

---

s_srcPathRoot

```
private static java.lang.String s_srcPathRoot
```

---

s_genSrcPath

```
private static java.lang.String s_genSrcPath
```

---

s_srcPkg

```
private static java.lang.String s_srcPkg
```

---

s_initDirCtx

```
private static javax.naming.directory.InitialDirContext s_initDirCtx
```

---

s_schemaRoot

```
private static javax.naming.directory.DirContext s_schemaRoot
```

---

s_dateGenerated

```
private static java.lang.String s_dateGenerated
```

---

s_copyRightYear

```
private static java.lang.String s_copyRightYear
```

Constructor Detail

Generator

```
public Generator()
```

Method Detail

main

```
public static void main(java.lang.String[] args)
```

Desc: Program entry point. If an object class name does not exist, an error message will be generated and processing continued.

Parameters:

`args` - Array of parameters. If the value of args[0].equals("ALL"), then interfaces/adapters will be generated for all object classes in the LDAP. Any further arguments are ignored. But if the value of args[0] is not equal to "ALL", then the arguments are expected to be LDAP object class names. Each name passed in will be processed and a resulting interface/adapter will be generated.

---

initialize

```
private static void initialize()
                    throws java.lang.Exception
```

Desc: Gets required system properties, figures out where to create the generated files.

---

generate

```
private static void generate(java.lang.String[] objClasses)
                     throws java.lang.Exception
```

Desc: Directs the generation of interface and adapter classes. Interfaces are prefixed with an "I" and adapters are suffixed with "Adapter". Also creates a file which contains constant strings used to identify object classes.

---

emitInterface

```
private static void emitInterface(java.lang.String oc,
                                  java.util.TreeSet[] attrNames,
                                  java.io.PrintStream out)
                           throws javax.naming.NamingException
```

Desc: Generates the interface class for the specified object class.

Parameters:

`oc` - Object Class to generate interface for.

`attrNames` - Array of TreeSet object containing the required and optional attribute names. Required is at index 0, optional at index 1.

`out` - Stream to write to.

---

emitAdapter

```
private static void emitAdapter(java.lang.String oc,
                                java.util.TreeSet[] attrNames,
                                java.io.PrintStream out)
                         throws javax.naming.NamingException
```

Desc: Generates the adapter class for the specified object class.
Parameters:
oc - Object Class to generate adapter for.
attrNames - Array of TreeSet object containing the required and optional attribute names. Required is at index 0, optional at index 1.
out - Stream to write to.

---

emitCommentHeader

```
private static void emitCommentHeader(java.io.PrintStream out)
```

Desc: Writes the comment header for the file.
Parameters:
out - Stream to write to.

---

emitPackage

```
private static void emitPackage(java.io.PrintStream out)
```

Desc: Writes the package statement.
Parameters:
out - Stream to write to.

---

emitInterfaceImports

```
private static void emitInterfaceImports(java.io.PrintStream out)
```

Desc: Writes the import statements for the Interface.
Parameters:
out - Stream to write to.

---

emitAdapterImports

```
private static void emitAdapterImports(java.io.PrintStream out)
```

Desc: Writes the import statements for the Adapter.
Parameters:
out - Stream to write to.

---

emitInterfaceName

```
private static void emitInterfaceName(java.lang.String oc,
                                      java.io.PrintStream out)
```

Desc: Writes the interface name and opening curley.
Parameters:
oc - Object class name.
out - Stream to write to.

---

emitAdapterName

```
private static void emitAdapterName(java.lang.String oc,
                                    java.io.PrintStream out)
```

Desc: Writes the adapter name and opening curley.
Parameters:
oc - Object class name.
out - Stream to write to.

---

emitDirConstName

```
private static void emitDirConstName(java.io.PrintStream out)
```

Desc: Writes the interface name and opening curley.
Parameters:
oc - Object class name.
out - Stream to write to.

---

emitInterfaceClassBody

```
private static void emitInterfaceClassBody(java.util.TreeSet[] attrNames,
                                           java.io.PrintStream out)
                                    throws javax.naming.NamingException
```

Desc: Writes the body of the interface.
Parameters:
attrNames - An array of TreeSet objects containing the required and optional object class attribute names.
TreeSet[0] = requried, TreeSet[1] = optional.
out - Stream to write to.

---

emitAdapterClassBody

```
private static void emitAdapterClassBody(java.lang.String oc,
                                         java.util.TreeSet[] attrNames,
                                         java.io.PrintStream out)
                                  throws javax.naming.NamingException
```

Desc: Writes the body of the adapter.
Parameters:
oc - Object class name.
attrNames - An array of TreeSet objects containing the required and optional object class attribute names.
TreeSet[0] = requried, TreeSet[1] = optional.
out - Stream to write to.

---

emitAllInterfaceGetters

```
private static void emitAllInterfaceGetters(java.util.TreeSet[] attrNames,
                                            java.io.PrintStream out)
```

Desc: Writes out all interface getters for both required and optional attributes.
Parameters:
[illegible] an array of TreeSet containing the required and optional attribute names.

out - Stream to write to.

---

emitAllInterfaceSetters

```
private static void emitAllInterfaceSetters(java.util.TreeSet[] attrNames,
                                            java.io.PrintStream out)
```

Desc: Writes out all interface setters for both required and optional attributes.
Parameters:
attrNames - An array of TreeSet containing the required and optional attribute names.
out - Stream to write to.

---

emitAllAdapterGetters

```
private static void emitAllAdapterGetters(java.util.TreeSet[] attrNames,
                                          java.io.PrintStream out)
```

Desc: Writes out all adapter getters for both required and optional attributes.
Parameters:
attrNames - An array of TreeSet containing the required and optional attribute names.
out - Stream to write to.

---

emitAllAdapterSetters

```
private static void emitAllAdapterSetters(java.util.TreeSet[] attrNames,
                                          java.io.PrintStream out)
```

Desc: Writes out all adapter setters for both required and optional attributes.
Parameters:
attrNames - An array of TreeSet containing the required and optional attribute names.
out - Stream to write to.

---

emitInterfaceGetters

```
private static void emitInterfaceGetters(java.util.TreeSet attrSet,
                                         java.io.PrintStream out)
```

Desc: For the specified 'attrSet' will generate getters for all attributes contained within for the interface.
Parameters:
attrSet - Set of attribute names to generate getters for.
out - Stream to write to.

---

emitInterfaceSetters

```
private static void emitInterfaceSetters(java.util.TreeSet attrSet,
                                         java.io.PrintStream out)
```

Desc: For the specified 'attrSet' will generate setters for all attributes contained within for the interface.
Parameters:
attrSet - Set of attribute names to generate setters for.
out - Stream to write to.

emitAdapterSetters

```
private static void emitAdapterSetters(java.util.TreeSet attrSet,
                                       java.io.PrintStream out)
```

Desc: Generate adapter setters for attributes.
Parameters:
attrSet - Set of attribute names to generate setters for.
out - Stream to write to.

---

emitAdapterGetters

```
private static void emitAdapterGetters(java.util.TreeSet attrSet,
                                       java.io.PrintStream out)
```

Desc: For the specified 'attrSet' will generate getters for all attributes contained within for the adapter.
Parameters:
attrSet - Set of attribute names to generate getters for.
out - Stream to write to.

---

emitGetFromModifiedCache

```
private static void emitGetFromModifiedCache(java.io.PrintStream out)
```

Desc: Writes out getFromModifiedCache method. This is a getter helper method which looks into the modified cache for changes.
Parameters:
out - Stream to write to.

---

emitAdapterToString

```
private static void emitAdapterToString(java.util.TreeSet[] attrNames,
                                        java.io.PrintStream out)
```

Desc: Writes out the adapters toString() method.
Parameters:
attrNames - An array of TreeSet containing the required and optional attribute names.
out - Stream to write to.

---

emitAdapterToStringHelper

```
private static void emitAdapterToStringHelper(java.io.PrintStream out)
```

Desc: Writes out the adapters toString() helper method.
Parameters:
out - Stream to write to.

---

emitDirConst

```
private static void emitDirConst(java.lang.String className,
                                 java.io.PrintStream out)
```

Desc: Write out adapter constant for the specified 'className'.
Parameters:
`className` - Object class name to write constant for.
`out` - Stream to write to.

---

emitClosingClassBracket

```
private static void emitClosingClassBracket(java.io.PrintStream out)
```

Desc: Writes the class closing curley.
Parameters:
`out` - Stream to write to.

---

shutDown

```
private static void shutDown()
                    throws javax.naming.NamingException
```

Desc: Release any remaing resources.

---

getObjClasses

```
private static java.lang.String[] getObjClasses()
                                        throws javax.naming.NamingException
```

Desc: This method will query LDAP to get all object class names and returns those names in an array of Strings.
Returns:
Returns an array of Strings containing all object class names.
Throws:
javax.naming.NamingException - If a naming exception occurs.

---

getAttrList

```
private static java.util.TreeSet[] getAttrList(java.lang.String oc)
                                        throws javax.naming.NamingException
```

Desc: Create a list of required and optional attribute names for the specified 'oc' object class and all attributes of 'oc' super object classes and so on by looking up these values in the LDAP Schema. Along with each attribute name is a boolean flag which indicates if the attribute is single valued or not. Returns this information in an array of TreeSet objects. The first element in the array contains the required attributes and the second element contains the optional attributes. The elements in contained in the TreeSet are Generator.NBP objects (Name Boolean Pair). The name is the attribute name and the boolean indicates if it is single valued or not.
Parameters:
oc - Object class name to build attribute list for.
Returns:
Array of TreeSet containing the required and optional attributes.

---

getAttributes

```
private static void getAttributes(java.lang.String oc,
                                  java.util.TreeSet mandatory,
                                  java.util.TreeSet optional)
                           throws javax.naming.NamingException
```

Recursively extracts all attributes for the specified 'oc' and all attributes of 'oc' superclasses.
Parameters:
`oc` - Object class name in which to extract attributes for.
`mandatory` - TreeSet to store mandatory attributes.
`optional` - TreeSet to store optional attributes.

---

addAttrNameToList

```
private static void addAttrNameToList(java.util.TreeSet store,
                                      javax.naming.NamingEnumeration vals)
                               throws javax.naming.NamingException
```

Desc: Add attribute names to a TreeSet. TreeSet does not allow dups. For each attribute 'vals', this method also determines if that attribute is single or multivalued. The attribute name and whether it is single valued or not is added to 'store' as Generator.NBP (Name Boolean Pair) object.
Parameters:
`store` - TreeSet to store attribute names
`vals` - Enumeration of attribute names

```
static void ()
```

Class Tree Deprecated Index Help
PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

com.geps.util.ldap

Class Generator.NBP

```
java.lang.Object
  |
  +--com.geps.util.ldap.Generator.NBP
```

Enclosing class:
Generator

---

private static class **Generator.NBP**
extends java.lang.Object
implements java.lang.Comparable

Desc: Helper class, Name Boolean Pair. Holds attributeName, isSingle boolean pair.

Field Summary

java.lang.String	**m_attrName**
boolean	**m_isSingle**

Constructor Summary

Generator.NBP(java.lang.String attrName, boolean isSingle)

Method Summary

int	**compareTo**(java.lang.Object obj)

Methods inherited from class java.lang.Object
, clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Field Detail

m_attrName

public java.lang.String **m_attrName**

---

m_isSingle

```
public boolean m_isSingle
```

Constructor Detail

Generator.NBP

```
public Generator.NBP(java.lang.String attrName,
                     boolean isSingle)
```

Method Detail

compareTo

```
public int compareTo(java.lang.Object obj)
```

Specified by:
compareTo in interface java.lang.Comparable

com.geps.util.ldap

Interface IBaseObjectClass

All Known Implementing Classes:
BaseDirectoryAdapter

---

public interface **IBaseObjectClass**

Class Description:

Base interface for all object class interfaces.

Method Summary

DirectoryEntry	**getDirEntry**()
java.util.ArrayList	**getModifications**()

Method Detail

getModifications

public java.util.ArrayList **getModifications**()

---

getDirEntry

public DirectoryEntry **getDirEntry**()

---

com.geps.util.ldap

Class Util

```
java.lang.Object
  |
  +--com.geps.util.ldap.Util
```

---

public abstract class Util
extends java.lang.Object

Class Description:

Utility methods used by classes in the com.geps.util.ldap package. This class is abstract and cannot be instantiated.

Field Summary

static java.lang.String	ADAPTER_SUFFIX
static java.lang.String	INTERFACE_PREFIX

Constructor Summary

Util()

Method Summary

static java.lang.String	convertToValidMethodName(java.lang.String str Desc: Will return a string with the same contents of 'str' out with the 1st character uppercased, the rest of the characters lower case and any '-' to '_'.

Methods inherited from class java.lang.Object
, clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Field Detail

INTERFACE_PREFIX

public static final java.lang.String **INTERFACE_PREFIX**

Appendix B

java.lang.reflect

Class Proxy

java.lang.Object
  |
  +--java.lang.reflect.Proxy

All Implemented Interfaces:
Serializable

---

public class **Proxy**
extends Object
implements Serializable

`Proxy` provides static methods for creating dynamic proxy classes and instances, and it is also the superclass of all dynamic proxy classes created by those methods

To create a proxy for some interface `Foo`.

```
InvocationHandler handler = new MyInvocationHandler(...);
Class proxyClass = Proxy.getProxyClass(
    Foo.class.getClassLoader(), new Class[] { Foo.class });
Foo f = (Foo) proxyClass.
    getConstructor(new Class[] { InvocationHandler.class }).
    newInstance(new Object[] { handler });
```

or more simply.

```
Foo f = (Foo) Proxy.newProxyInstance(Foo.class.getClassLoader(),
                                     new Class[] { Foo.class },
                                     handler);
```

A *dynamic proxy class* (simply referred to as a *proxy class* below) is a class that implements a list of interfaces specified at runtime when the class is created, with behavior as described below. A *proxy interface* is such an interface that is implemented by a proxy class. A *proxy instance* is an instance of a proxy class. Each proxy instance has an associated *invocation handler* object, which implements the interface `InvocationHandler`. A method invocation on a proxy instance through one of its proxy interfaces will be dispatched to the `invoke` method of the instance's invocation handler, passing the proxy instance, a `java.lang.reflect.Method` object identifying the method that was invoked, and an array of type `Object` containing the arguments. The invocation handler processes the encoded method invocation as appropriate and the result that it returns will be returned as the result of the

method invocation on the proxy instance.

A proxy class has the following properties

- Proxy classes are public, final, and not abstract
- The unqualified name of a proxy class is unspecified The space of class names that begin with the string "$Proxy" should be, however, reserved for proxy classes
- A proxy class extends java.lang.reflect.Proxy.
- A proxy class implements exactly the interfaces specified at its creation, in the same order
- If a proxy class implements a non-public interface, then it will be defined in the same package as that interface Otherwise, the package of a proxy class is also unspecified. Note that package sealing will not prevent a proxy class from being successfully defined in a particular package at runtime, and neither will classes already defined in the same class loader and the same package with particular signers.
- Since a proxy class implements all of the interfaces specified at its creation, invoking getInterfaces on its Class object will return an array containing the same list of interfaces (in the order specified at its creation), invoking getMethods on its Class object will return an array of Method objects that include all of the methods in those interfaces, and invoking getMethod will find methods in the proxy interfaces as would be expected
- The Proxy.isProxyClass method will return true if it is passed a proxy class-- a class returned by Proxy.getProxyClass or the class of an object returned by Proxy.newProxyInstance-- and false otherwise
- The java.security.ProtectionDomain of a proxy class is the same as that of system classes loaded by the bootstrap class loader, such as java.lang.Object, because the code for a proxy class is generated by trusted system code This protection domain will typically be granted java.security.AllPermission
- Each proxy class has one public constructor that takes one argument, an implementation of the interface InvocationHandler, to set the invocation handler for a proxy instance. Rather than having to use the reflection API to access the public constructor, a proxy instance can be also be created by calling the Proxy.newInstance method, which combines the actions of calling Proxy.getProxyClass with invoking the constructor with an invocation handler

A proxy instance has the following properties

- Given a proxy instance proxy and one of the interfaces implemented by its proxy class Foo, the following expression will return true:

```
proxy instanceof Foo
```

and the following cast operation will succeed (rather than throwing a ClassCastException):

```
(Foo) proxy
```

- Each proxy instance has an associated invocation handler, the one that was passed to its constructor. The static Proxy.getInvocationHandler method will return the invocation handler associated with the proxy instance passed as its argument.
- An interface method invocation on a proxy instance will be encoded and dispatched to the invocation handler's invoke method as described in the documentation for that method.
- An invocation of the hashCode, equals, or toString methods declared in java.lang.Object on a proxy

instance will be encoded and dispatched to the invocation handler's `invoke` method in the same manner as interface method invocations are encoded and dispatched, as described above The declaring class of the `Method` object passed to `invoke` will be `java.lang.Object` Other public methods of a proxy instance inherited from `java.lang.Object` are not overridden by a proxy class, so invocations of those methods behave like they do for instances of `java.lang.Object`.

Methods Duplicated in Multiple Proxy Interfaces

When two or more interfaces of a proxy class contain a method with the same name and parameter signature, the order of the proxy class's interfaces becomes significant When such a *duplicate method* is invoked on a proxy instance, the `Method` object passed to the invocation handler will not necessarily be the one whose declaring class is assignable from the reference type of the interface that the proxy's method was invoked through This limitation exists because the corresponding method implementation in the generated proxy class cannot determine which interface it was invoked through Therefore, when a duplicate method is invoked on a proxy instance, the `Method` object for the method in the foremost interface that contains the method (either directly or inherited through a superinterface) in the proxy class's list of interfaces is passed to the invocation handler's `invoke` method, regardless of the reference type through which the method invocation occurred

If a proxy interface contains a method with the same name and parameter signature as the `hashCode`, `equals`, or `toString` methods of `java.lang.Object`, when such a method is invoked on a proxy instance, the `Method` object passed to the invocation handler will have `java.lang.Object` as its declaring class. In other words, the public, non-final methods of `java.lang.Object` logically precede all of the proxy interfaces for the determination of which `Method` object to pass to the invocation handler

Note also that when a duplicate method is dispatched to an invocation handler, the `invoke` method may only throw checked exception types that are assignable to one of the exception types in the `throws` clause of the method in *all* of the proxy interfaces that it can be invoked through If the `invoke` method throws a checked exception that is not assignable to any of the exception types declared by the method in one of the the proxy interfaces that it can be invoked through, then an unchecked `UndeclaredThrowableException` will be thrown by the invocation on the proxy instance This restriction means that not all of the exception types returned by invoking `getExceptionTypes` on the `Method` object passed to the `invoke` method can necessarily be thrown successfully by the `invoke` method

Since:

JDK1.3

See Also:

`InvocationHandler`, Serialized Form

Field Summary

protected InvocationHandler	**h** the invocation handler for this proxy instance

Constructor Summary

protected	**Proxy**(InvocationHandler h) Constructs a new Proxy instance from a subclass (typically, a dynamic proxy class) with the specified value for its invocation handler

Method Summary

static InvocationHandler	**getInvocationHandler**(Object proxy) Returns the invocation handler for the specified proxy instance
static Class	**getProxyClass**(ClassLoader loader, Class[] interfaces) Returns the java.lang.Class object for a proxy class given a class loader and an array of interfaces
static boolean	**isProxyClass**(Class cl) Returns true if and only if the specified class was dynamically generated to be a proxy class using the getProxyClass method or the newProxyInstance method
static Object	**newProxyInstance**(ClassLoader loader, Class[] interfaces, InvocationHandler h) Returns an instance of a proxy class for the specified interfaces that dispatches method invocations to the specified invocation handler

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

h

```
protected InvocationHandler h
```

the invocation handler for this proxy instance.

Constructor Detail

Proxy

```
protected Proxy(InvocationHandler h)
```

Constructs a new Proxy instance from a subclass (typically, a dynamic proxy class) with the specified value for its invocation handler.

Parameters:

`h` - the invocation handler for this proxy instance

Method Detail

getProxyClass

```
publ.  static Class getProxyClass(ClassLoader loader,
                                  Class[] interfaces)
                           throws IllegalArgumentException
```

Returns the `java.lang.Class` object for a proxy class given a class loader and an array of interfaces The proxy class will be defined in the specified class loader and will implement all of the supplied interfaces If a proxy class for the same permutation of interfaces has already been defined in the class loader, then the existing proxy class will be returned, otherwise, a proxy class for those interfaces will be generated dynamically and defined in the class loader.

There are several restrictions on the parameters that may be passed to `Proxy.getProxyClass`

- All of the `Class` objects in the `interfaces` array must represent interfaces, not classes or primitive types
- No two elements in the `interfaces` array may refer to identical `Class` objects
- All of the interface types must be visible by name through the specified class loader In other words, for class loader `cl` and every interface `i`, the following expression must be true

```
Class.forName(i.getName(), false, cl  == i
```

- All non-public interfaces must be in the same package, otherwise, it would not be possible for the proxy class to implement all of the interfaces, regardless of what package it is defined in
- No two interfaces may each have a method with the same name and parameter signature but different return type
- The resulting proxy class must not exceed any limits imposed on classes by the virtual machine For example, the VM may limit the number of interfaces that a class may implement to 65535, in that case, the size of the `interfaces` array must not exceed 65535

If any of these restrictions are violated, `Proxy.getProxyClass` will throw an `IllegalArgumentException`. If the `interfaces` array argument or any of its elements are `null`, a `NullPointerException` will be thrown

Note that the order of the specified proxy interfaces is significant. two requests for a proxy class with the same combination of interfaces but in a different order will result in two distinct proxy classes.

Parameters:

`loader` - the class loader to define the proxy class in
`interfaces` - the list of interfaces for the proxy class to implement

Returns:

a proxy class that is defined in the specified class loader and that implements the specified interfaces

Throws:

IllegalArgumentException - if any of the restrictions on the parameters that may be passed to getProxyClass are violated
NullPointerException - if the interfaces array argument or any of its elements are null

---

newProxyInstance

```
public static Object newProxyInstance(ClassLoader loader,
                                       Class[] interfaces,
                                       InvocationHandler h)
                                throws IllegalArgumentException
```

Returns an instance of a proxy class for the specified interfaces that dispatches method invocations to the specified invocation handler. This method is equivalent to

```
Proxy.getProxyClass(loader, interfaces).
    getConstructor(new Class[] { InvocationHandler.class }).
    newInstance(new Object[] { handler });
```

Proxy.newProxyInstance throws IllegalArgumentException for the same reasons that Proxy.getProxyClass does

Parameters:

loader - the class loader to define the proxy class in
interfaces - the list of interfaces for the proxy class to implement
h - the invocation handler to dispatch method invocations to

Returns:

a proxy instance with the specified invocation handler of a proxy class that is defined in the specified class loader and that implements the specified interfaces

Throws:

IllegalArgumentException - if any of the restrictions on the parameters that may be passed to getProxyClass are violated
NullPointerException - if the interfaces array argument or any of its elements are null, or if the invocation handler, h, is null

---

isProxyClass

```
public static boolean isProxyClass(Class cl)
```

Returns true if and only if the specified class was dynamically generated to be a proxy class using the getProxyClass method or the newProxyInstance method

The reliability of this method is important for the ability to use it to make security decisions, so its implementation should not just test if the class in question extends Proxy.

Parameters:

`cl` - the class to test

Returns:

`true` if the class is a proxy class and `false` otherwise

Throws:

`NullPointerException` - if `cl` is `null`

---

getInvocationHandler

```
public static InvocationHandler getInvocationHandler(Object proxy)
                                              throws IllegalArgumentException
```

Returns the invocation handler for the specified proxy instance.

Parameters:

`proxy` - the proxy instance to return the invocation handler for

Returns:

the invocation handler for the proxy instance

Throws:

`IllegalArgumentException` - if the argument is not a proxy instance

---

java.lang.reflect

Interface InvocationHandler

public interface **InvocationHandler**

`InvocationHandler` is the interface implemented by the *invocation handler* of a proxy instance

Each proxy instance has an associated invocation handler When a method is invoked on a proxy instance, the method invocation is encoded and dispatched to the `invoke` method of its invocation handler

Since:

JDK1 3

See Also:

`Proxy`

Method Summary

Object	**invoke**(Object proxy, Method method, Object[] args) Processes a method invocation on a proxy instance and returns the result

Method Detail

invoke

```
public Object invoke(Object proxy,
                     Method method,
                     Object[] args)
              throws Throwable
```

Processes a method invocation on a proxy instance and returns the result. This method will be invoked on an invocation handler when a method is invoked on a proxy instance that it is associated with.

Parameters:

`proxy` - the proxy instance that the method was invoked on

`method` - the `Method` instance corresponding to the interface method invoked on the proxy instance The declaring class of the `Method` object will be the interface that the method was declared in, which may be a superinterface of the proxy interface that the proxy class inherits the method through.

`args` - an array of objects containing the values of the arguments passed in the method invocation on the proxy instance, or `null` if interface method takes no arguments. Arguments of primitive types are wrapped in instances of the appropriate primitive wrapper class, such as `java.lang.Integer` or `java.lang.Boolean`

Returns:

the value to return from the method invocation on the proxy instance If the declared return type of the interface method is a primitive type, then the value returned by this method must be an instance of the corresponding primitive wrapper class, otherwise, it must be a type assignable to the declared return type If the value returned by this method is `null` and the interface method's return type is primitive, then a `NullPointerException` will be thrown by the method invocation on the proxy instance If the value returned by this method is otherwise not compatible with the interface method's declared return type as described above, a `ClassCastException` will be thrown by the method invocation on the proxy instance

Throws:

`Throwable` - the exception to throw from the method invocation on the proxy instance The exception's type must be assignable either to any of the exception types declared in the `throws` clause of the interface method or to the unchecked exception types `java.lang.RuntimeException` or `ava.lang.Error`. If a checked exception is thrown by this method that is not assignable to any of the exception types declared in the `throws` clause of the interface method, then an `UndeclaredThrowableException` containing the exception that was thrown by this method will be thrown by the method invocation on the proxy instance

See Also:

`UndeclaredThrowableException`

For further API reference and developer documentation, see Java 2 SDK SE Developer Documentation That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Appendix C

Package com.gepower.sfo.tool.ldap

Interface Summary	
DirectoryManager	DirectoryManager provides an interface for accessing Directory data
DirectorySource	DirectorySource is an interface which provides access to Directory data sources

Class Summary	
DefaultDirectorySource	DirectorySource implementation.
DirectoryEntry	Represents an LDAP Directory Entry and a LDAP invocation handler used in Proxy instances
DirectoryManagerFactory	Use to create an object which implements the DirectoryManager interface
Generator	Generates java interfaces which represents LDAP object classes

Package com.gepower.sfo.tool.ldap

Interface Summary	
DirectoryManager	DirectoryManager provides an interface for accessing Directory data
DirectorySource	DirectorySource is an interface which provides access to Directory data sources

Class Summary	
DefaultDirectorySource	DirectorySource implementation.
DirectoryEntry	Represents an LDAP Directory Entry and a LDAP invocation handler used in Proxy instances
DirectoryManagerFactory	Use to create an object which implements the DirectoryManager interface.
Generator	Generates java interfaces which represents LDAP object classes

com.gepower.sfo.tool.ldap

Interface DirectoryManager

public interface **DirectoryManager**

Dire ryManager provides an interface for accessing Directory data.

Method Summary

[illegible]	**cast**(java.lang.Object entry, java.lang.Class interfaceToCastTo) This method provides backward compatibility from Java 1.3 to Java 1 2 which does not support the Proxy class
[illegible]	**getDN**(java.lang.Object entry) Use to get the specified 'entry' distinquished name.
[illegible]	**lookup**(java.lang.String dn) Use to lookup a specific entry identified by the specified 'dn'.
[illegible]	**mixinInterfaces**(java.lang.Object entry, java.lang.Class[] newInterfaces) Use to add additional interfaces specified by 'newInterfaces' to an existing 'entry'
[illegible]	**newEntryInstance**(java.lang.String dn, java.lang.Class[] interfaces) Use to create a new LDAP entry
[illegible]	**remove**(java.lang.Object entry) Use to remove an existing entry from the Directory.
[illegible]	**search**(java.lang.String ctxToSearch, java.lang.String filter) Use to execute a query against the Directory using the specified 'ctxToSearch', and 'filter'.
[illegible]	**search**(java.lang.String ctxToSearch, java.lang.String filter, javax.naming.directory.SearchControls searchCtrls) Use to execute a query against the Directory using the specified 'ctxToSearch', 'filter', and 'searchCtrls.
[illegible]	**search**(java.lang.String ctxToSearch, java.lang.String filter, javax.naming.directory.SearchControls searchCtrls, javax.naming.ldap.Control[] reqCtrls) Use to execute a query against the Directory using the specified 'ctxToSearch', 'filter', 'searchCtrls, and 'reqCtrls'

	write(java.lang.Object entry) Use to commit a new entry or modifications of an existing entry to the Directory

Method Detail

newEntryInstance

```
public java.lang.Object newEntryInstance(java.lang.String dn,
                                         java.lang.Class[] interfaces)
                                  throws javax.naming.NamingException
```

Use to create a new LDAP entry The entry is not written to the Directory until DirectoryManager write() is executed

Parameters:

dn - Distinquished name for the new entry Must not be null or empty

interfaces - Array of Class objects which represent the interfaces that this new entry will support The Class objects MUST be one of the LDAP code generated interfaces. Array must not be null or empty

Returns:

Object representing the directory entry This object can be cast to the appropriate "objectclass" interface (s)

Throws:

javax.naming.NamingException - if a naming exception is encountered

---

mixinInterfaces

```
public java.lang.Object mixinInterfaces(java.lang.Object entry,
                                        java.lang.Class[] newInterfaces)
                                 throws javax.naming.NamingException
```

Use to add additional interfaces specified by 'newInterfaces' to an existing 'entry'. The modified entry is not written to the Directory until DirectoryManager.write() is executed.

Parameters:

entry - Existing LDAP entry to mix new interfaces into 'entry' must be acquired by calls to DirectoryController.lookup(), DirectoryController search(), or DirectoryController newEntryInstance()

newInterfaces - Array of new interfaces to mix into the entry Must not be null and must not be empty.

Returns:

Object representing the modified directory entry. This object can be cast to the appropriate "objectclass" interface(s) including those contained in 'newInterfaces'

Throws:

javax.naming.NamingException - if a naming exception is encountered

---

lookup

```
public java.lang.Object lookup(java.lang.String dn)
                         throws javax.naming.NameNotFoundException,
                                javax.naming.NamingException,
                                java.lang.ClassNotFoundException
```

Use to lookup a specific entry identified by the specified 'dn'

Parameters:

dn - The distinquished name which uniquely identifies the entry Must not be null and must not be empty

Returns:

Object representing the directory entry bound to the specified dn This object can be cast to the appropriate "objectclass" interface(s)

Throws:

javax.naming.NameNotFoundException - if dn cannot be resolved because it is not bound

javax.naming.NamingException - if a naming exception is encountered.

java.lang.ClassNotFoundException - if the looked up entry contains an object class which does not have an associated code generated interface

---

search

```
public java.util.List search(java.lang.String ctxToSearch,
                             java.lang.String filter)
                      throws javax.naming.NamingException,
                             java.lang.ClassNotFoundException
```

Use to execute a query against the Directory using the specified 'ctxToSearch', and 'filter'

Parameters:

ctxToSearch - Context to search "" for current context Must not be null

filter - LDAP search filter Must not be null

Returns:

List of Objects representing the results of the search These Object can each be cast to the appropriate "objectclass" interface(s). If search finds nothing, List returned will have size of zero. Return will never be null

Throws:

java.lang.ClassNotFoundException - if the entries found contains an object class which does not have an associated code generated interface.

javax.naming.NamingException - if naming exception is encountered.

---

search

```
public java.util.List search(java.lang.String ctxToSearch,
                             java.lang.String filter,
                             javax.naming.directory.SearchControls searchCtrls)
                      throws javax.naming.NamingException,
                             java.lang.ClassNotFoundException
```

Use to execute a query against the Directory using the specified 'ctxToSearch', 'filter', and 'searchCtrls

Parameters:

`ctxToSearch` - Context to search "" for current context. Must not be null

`filter` - LDAP search filter Must not be null

`searchCtrls` - Used to determine scope of search and what gets returned May be null If null, defaults will be used (search using SearchControls SUBTREE_SCOPE)

Returns:

List of Objects representing the results of the search These Object can each be cast to the appropriate "objectclass" interface(s). If search finds nothing, List returned will have size of zero Return will never be null.

Throws:

`java.lang.ClassNotFoundException` - if the entries found contains an object class which does not have an associated code generated interface

`javax.naming.NamingException` - if naming exception is encountered.

See Also:

`SearchControls`

---

search

```
public java.util.List search(java.lang.String ctxToSearch,
                             java.lang.String filter,
                             javax.naming.directory.SearchControls searchCtrls,
                             javax.naming.ldap.Control[] reqCtrls)
                      throws javax.naming.NamingException,
                             java.lang.ClassNotFoundException
```

Use to execute a query against the Directory using the specified 'ctxToSearch', 'filter', 'searchCtrls, and 'reqCtrls'

Parameters:

`ctxToSearch` - Context to search "" for current context. Must not be null.

`filter` - LDAP search filter. Must not be null.

`searchCtrls` - Used to determine scope of search and what gets returned May be null. If null, defaults will be used (search using SearchControls.SUBTREE_SCOPE)

`reqCtrls` - A control to request the LDAP search to return in a certain way (i.e, sort results in a particular way) May be null. If null, no LDAP request controls will be used

Returns:

List of Objects representing the results of the search. These Object can each be cast to the appropriate "objectclass" interface(s). If search finds nothing, List returned will have size of zero. Return will never be null.

Throws:

`java.lang.ClassNotFoundException` - if the entries found contains an object class which does not have an associated code generated interface.

`javax.naming.NamingException` - if naming exception is encountered

See Also:

`SearchControls, Control`

---

write

```
public void write(java.lang.Object entry)
          throws javax.naming.NamingException
```

Use to commit a new entry or modifications of an existing entry to the Directory

Parameters:

`entry` - Entry to commit to the directory 'entry' must have been acquired by calls to DirectoryController lookup(), DirectoryController search(), or DirectoryController newEntryInstance() Must not be null

Throws:

`javax.naming.NamingException` - if naming exception is encountered.

---

remove

```
public void remove(java.lang.Object entry)
            throws javax.naming.NamingException
```

Use to remove an existing entry from the Directory.

Parameters:

`entry` - Entry to remove from the directory 'entry' must have been acquired by calls to DirectoryController lookup(), DirectoryController search(), or DirectoryController newEntryInstance() Must not be null

Throws:

`javax.naming.NamingException` - if naming exception is encountered

---

cast

```
public java.lang.Object cast(java.lang.Object entry,
                             java.lang.Class interfaceToCastTo)
                      throws java.lang.ClassCastException
```

This method provides backward compatibility from Java 1.3 to Java 1.2 which does not support the Proxy class This method is not yet implemented

Parameters:

`entry` - Entry to cast. 'entry' must have been acquired by calls to DirectoryController.lookup(), DirectoryController.search(), or DirectoryController newEntryInstance() Must not be null

`interfaceToCastTo` - This is the interface that the specfied 'entry' is to be cast to.

Returns:

Object which can be cast to the type specified by 'interfaceToCastTo'

Throws:

`java.lang.ClassCastException` - if the specified 'entry' cannot be cast to the specified

'interfaceToCastTo'

---

getDN

```
public java.lang.String getDN(java.lang.Object entry)
```

Use to get the specified 'entry' distinquished name.

Parameters:

`entry` - Entry to obtain distinquished name from 'entry' must have been acquired by calls to DirectoryController.lookup(), DirectoryController.search(), or DirectoryController newEntryInstance() Must not be null

Returns:

String containing the specified 'entry' distinquished name.

---

com.gepower.sfo.tool.ldap

Class DefaultDirectorySource

```
java.lang.Object
  +--com.gepower.sfo.tool.ldap.DefaultDirectorySource
```

All Implemented Interfaces:
DirectorySource

---

public class **DefaultDirectorySource**
extends java.lang.Object
implements DirectorySource

DirectorySource implementation. See DirectorySource for discription of implemented methods

---

Constructor Summary

DefaultDirectorySource(java.util.Hashtable environment)

Method Summary

void	**discardDirContext**(javax.naming.directory.DirContext context) Use to discard the specified 'context'.
javax.naming.directory.DirContext	**getDirContext**() Use to get a JNDI DirContext object.
void	**releaseDirContext**(javax.naming.directory.DirContext context) Use to release the specified 'context'.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

DefaultDirectorySource

```
public DefaultDirectorySource(java.util.Hashtable environment
                        throws javax.naming.NamingException
```

Method Detail

getDirContext

```
public javax.naming.directory.DirContext getDirContext()
                                                  throws javax.naming.NamingException
```

Description copied from interface: `DirectorySource`
Use to get a JNDI DirContext object
Specified by:
`getDirContext` in interface `DirectorySource`
Following copied from interface `com.gepower.sfo.tool.ldap.DirectorySource`
Returns:
DirContext object
Throws:
`javax.naming.NamingException` - if a naming exception is encountered.

---

releaseDirContext

```
public void releaseDirContext(javax.naming.directory.DirContext context)
```

Description copied from interface: `DirectorySource`
Use to release the specified 'context' This should be called when the context is no longer needed.
Specified by:
`releaseDirContext` in interface `DirectorySource`
Following copied from interface `com.gepower.sfo.tool.ldap.DirectorySource`
Parameters:
`context` - The context to release

---

discardDirContext

```
public void discardDirContext(javax.naming.directory.DirContext context)
```

Description copied from interface: `DirectorySource`
Use to discard the specified 'context'
Specified by:
`discardDirContext` in interface `DirectorySource`

Following copied from interface: `com.gepower.sfo.tool.ldap.DirectorySource`

Parameters:

`context` - The context to release

com.gepower.sfo.tool.ldap

Class DirectoryEntry

```
java.lang.Object
  |
  +--com.gepower.sfo.tool.ldap.DirectoryEntry
```

All Implemented Interfaces:

java lang.reflect InvocationHandler, java io.Serializable

---

public class **DirectoryEntry**
extends java lang Object
implements java io Serializable, java lang reflect.InvocationHandler

Represents an LDAP Directory Entry and a LDAP invocation handler used in Proxy instances Each proxy instance has an associated invocation handler. When a method is invoked on a proxy instance, the method invocation is encoded and dispatched to the invoke method of its invocation handler. This is a package scope class and not used directly by clients

See Also:

InvocationHandler, java.lang.reflect.Proxy, Serialized Form

Method Summary

java.lang.Object	**invoke**(java.lang.Object proxy, java.lang.reflect.Method method, java.lang.Object[] args) Implement abstract method invoke() from InvocationHandler
java.lang.String	**toString**(java.lang.Object proxy, java.lang.reflect.Method method) Returns the contents of all attribute in this entry

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Method Detail

invoke

```
public java.lang.Object invoke(java.lang.Object proxy,
                               java.lang.reflect.Method method,
                               java.lang.Object[] args)
                        throws java.lang.Throwable
```

Implement abstract method invoke() from InvocationHandler This method only recognizes methods that have been declared in the generated interfaces.

Specified by:

`invoke` in interface `java.lang.reflect.InvocationHandler`

Parameters:

`proxy` - the proxy instance that the method was invoked on.

`method` - the Method instance corresponding to the interface method invoked on the proxy instance The declaring class of the Method object will be the interface that the method was declared in, which may be a superinterface of the proxy interface that the proxy class inherits the method through.

`args` - an array of objects containing the values of the arguments passed in the method invocation on the proxy instance, or null if interface method takes no arguments. Arguments of primitive types are wrapped in instances of the appropriate primitive wrapper class, such as java.lang.Integer or java lang.Boolean.

Throws:

`java.lang.Throwable` - the exception to throw from the method invocation on the proxy instance The exception's type must be assignable either to any of the exception types declared in the throws clause of the interface method or to the unchecked exception types java lang RuntimeException or java lang Error If a checked exception is thrown by this method that is not assignable to any of the exception types declared in the throws clause of the interface method, then an UndeclaredThrowableException containing the exception that was thrown by this method will be thrown by the method invocation on the proxy instance

See Also:

`java.lang.reflect.UndeclaredThrowableException`

---

toString

```
public java.lang.String toString(java.lang.Object proxy,
                                 java.lang.reflect.Method method)
```

Returns the contents of all attribute in this entry Use for debugging purposes only

Parameters:

`proxy` - The Proxy object serviced by this InvocationHandler.

`method` - The Method object invoked on the Proxy

Returns:

The contents of all attributes in this entry.

---

com.gepower.sfo.tool.ldap

Class DirectoryManagerFactory

```
java.lang.Object
  ---com.gepower.sfo.tool.ldap.DirectoryManagerFactory
```

public class **DirectoryManagerFactory**
extends java.lang.Object

Use to create an object which implements the DirectoryManager interface.

Constructor Summary

DirectoryManagerFactory()

Method Summary

static DirectoryManager	**newDirectoryManager**(DirectorySource src, java.lang.String pkg) Creates a new object which implements the DirectoryManager interface using the specified 'src' and 'pkg'.
static DirectoryManager	**newDirectoryManager**(DirectorySource src, java.lang.String pkg, java.lang.ClassLoader loader) Creates a new object which implements the DirectoryManager interface using the specified 'src', 'pkg', and 'loader'.
static DirectoryManager	**newDirectoryManager**(DirectorySource src, java.lang.String pkg, java.lang.ClassLoader loader, java.io.PrintStream logger) Creates a new object which implements the DirectoryManager interface using the specified 'src', 'pkg', 'loader', and 'logger'
static DirectoryManager	**newDirectoryManager**(java.util.Hashtable env, java.lang.String pkg) Creates a new object which implements the DirectoryManager interface using the specified 'env' and 'pkg'

static DirectoryManager	**newDirectoryManager**(java.util.Hashtable env, java.lang.String pkg, java.lang.ClassLoader loader) Creates a new object which implements the DirectoryManager interface using the specified 'env', 'pkg', and 'loader'
static DirectoryManager	**newDirectoryManager**(java.util.Hashtable env, java.lang.String pkg, java.lang.ClassLoader loader, java.io.PrintStream logger) Creates a new object which implements the DirectoryManager interface using the specified 'env', 'pkg', 'loader', and 'logger'.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

DirectoryManagerFactory

```
public DirectoryManagerFactory()
```

Method Detail

newDirectoryManager

```
public static DirectoryManager newDirectoryManager(java.util.Hashtable env,
                                                   java.lang.String pkg)
                                            throws java.lang.IllegalArgumentException,
                                                   javax.naming.NamingException
```

Creates a new object which implements the DirectoryManager interface using the specified 'env' and 'pkg'

Parameters:

`env` - Used to specify various preferences and properties that define the environment in which naming and directory services are accessed. Must not be null.

`pkg` - The java package in which the LDAP interfaces were generated under. Must not be null.

Throws:

`java.lang.IllegalArgumentException` - if 'env' or 'pkg' is null.

`javax.naming.NamingException` - if a naming exception is encountered.

---

newDirectoryManager

```
public static DirectoryManager newDirectoryManager(java.util.Hashtable env,
                                                   java.lang.String pkg,
                                                   java.lang.ClassLoader loader)
                                            throws java.lang.IllegalArgumentException,
```

javax.naming.NamingException

Creates a new object which implements the DirectoryManager interface using the specified 'env', 'pkg', and 'loader'

Parameters:

env - Used to specify various preferences and properties that define the environment in which naming and directory services are accessed Must not be null

pkg - The java package in which the LDAP interfaces were generated under Must not be null

loader - Class loader to use to load proxy classes May be null in which case the current threads class loader will be used

Throws:

java.lang.IllegalArgumentException - if 'env' or 'pkg' is null.

javax.naming.NamingException - if a naming exception is encountered.

---

newDirectoryManager

```
public static DirectoryManager newDirectoryManager(java.util.Hashtable env,
                                                    java.lang.String pkg,
                                                    java.lang.ClassLoader loader,
                                                    java.io.PrintStream logger)
                                             throws java.lang.IllegalArgumentException,
                                                    javax.naming.NamingException
```

Creates a new object which implements the DirectoryManager interface using the specified 'env', 'pkg', 'loader', and 'logger'.

Parameters:

env - Used to specify various preferences and properties that define the environment in which naming and directory services are accessed Must not be null

pkg - The java package in which the LDAP interfaces were generated under Must not be null

loader - Class loader to use to load proxy classes. May be null in which case the current threads class loader will be used

logger - This is where all debug trace messages will be written to.

Throws:

java.lang.IllegalArgumentException - if 'env' or 'pkg' is null

javax.naming.NamingException - if a naming exception is encountered.

---

newDirectoryManager

```
public static DirectoryManager newDirectoryManager(DirectorySource src,
                                                    java.lang.String pkg)
                                             throws java.lang.IllegalArgumentException,
                                                    javax.naming.NamingException
```

Creates a new object which implements the DirectoryManager interface using the specified 'src' and 'pkg'.

Parameters:

src - Specifies the what directory source the DirectoryManager will use. Must not be null.

pkg - The java package in which the LDAP interfaces were generated under Must not be null

Throws:

java.lang.IllegalArgumentException - if 'src' or 'pkg' is null.

javax.naming.NamingException - if a naming exception is encountered.

---

newDirectoryManager

```
public static DirectoryManager newDirectoryManager(DirectorySource src,
                                                    java.lang.String pkg,
                                                    java.lang.ClassLoader loader
                                                  throws java.lang.IllegalArgumentException,
                                                    javax.naming.NamingException
```

Creates a new object which implements the DirectoryManager interface using the specified 'src'. 'pkg'. and 'loader'

Parameters:

src - Specifies the what directory source the DirectoryManager will use. Must not be null.

pkg - The java package in which the LDAP interfaces were generated under Must not be null.

loader - Class loader to use to load proxy classes. May be null in which case the current threads class loader will be used.

Throws:

java.lang.IllegalArgumentException - if 'src' or 'pkg' is null.

javax.naming.NamingException - if a naming exception is encountered

---

newDirectoryManager

```
public static DirectoryManager newDirectoryManager(DirectorySource src,
                                                    java.lang.String pkg,
                                                    java.lang.ClassLoader loader,
                                                    java.io.PrintStream logger)
                                                  throws java.lang.IllegalArgumentException,
                                                    javax.naming.NamingException
```

Creates a new object which implements the DirectoryManager interface using the specified 'src', 'pkg', 'loader', and 'logger'

Parameters:

src - Specifies the what directory source the DirectoryManager will use. Must not be null.

pkg - The java package in which the LDAP interfaces were generated under. Must not be null.

loader - Class loader to use to load proxy classes. May be null in which case the current threads class loader will be used.

logger - This is where all debug trace messages will be written to.

Throws:

java.lang.IllegalArgumentException - if 'src' or 'pkg' is null.

javax.naming.NamingException - if a naming exception is encountered

---

com.gepower.sfo.tool.ldap

Class Generator

```
java.lang.Object
   |
   +--com.gepower.sfo.tool.ldap.Generator
```

public abstract class **Generator**
extends java.lang.Object

Generates java interfaces which represents LDAP object classes. These classes are used in the java LDAP Directory framework. This class is abstract can contains only static methods. This class contains a main() method and is designed to be executed from the command line See method description for main() for more details

See Also:
main(java.lang.String[])

Constructor Summary

Generator()

Method Summary

static void	main(java.lang.String[] args) Usage: java com.gepower.sfo tool.ldap.Generator params [options]

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Generator

```
public Generator()
```

Method Detail

main

```
public static void main(java.lang.String[] args)
```

Usage: java com.gepower.sfo.tool.ldap Generator params [options]

To print out help, use the -help option when executing this program from the command line

Parameters:

`args` - Array of String arguments which consists of the required and optional parameters

Required Parameters

- '-sourcerootpath' the root directory path for the generated java source
- '-package' the java package for the generated java source
- '-dirctxfactory' class to use for the initial directory context factory
- '-providerurl' the LDAP URL string (i.e , ldap //localhost.389/o=ge com)
- '-securityprincipal' identity of the principal for authenticating the caller to the service
- '-securitycredentials' credentials of the principal for authenticating the caller to the service
- '-securityauthentication' security level to use

Optional Parameters:

- '-exclude' object classes matching the wildcard will be excluded from code generation Exclusions have precedence over Inclusion. Multiple wildcards can be specified separated by semi-colons (i e. "ns*; ob*; net*server")
- '-include' object classes matching the wildcard will be included in code generation If option not specified, include all object classes Multiple wildcards can be specified, see exclude option
- '-version' version number that will be included into the javadoc of the generated code
- '-tabstop' tab stop to use when formatting the generated code
- '-help' use to print usage syntax on the command line
- '-?' use to print usage syntax on the command line